lab6-lanston

November 27, 2023

[6]:
```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

[7]:
```
df = spark.read.load("s3://lanston-bigdata-lab6/lab6/data.csv", format="csv",␣
↪sep=",", inferSchema="true", header="true")
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

[8]:
```
df.write.option("path", "/user/hadoop/nasa")   #save it as csv
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...

<pyspark.sql.readwriter.DataFrameWriter object at 0x7f666b5b7890>
```

[13]:
```
df.write.format("parquet").option("path", "/user/hadoop/lab6_parquet").save()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

[14]:
```
# read it as parquet
parquet_path = "/user/hadoop/lab6_parquet"
df_parquet = spark.read.parquet(parquet_path)
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

[16]:
```
df.createOrReplaceTempView("web_logs")
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

[17]:
```
import time

start_time = time.time()
top_hosts_by_url_count = spark.sql("""
SELECT host, COUNT(url) as cnt
FROM web_logs
GROUP BY host
ORDER BY cnt DESC
LIMIT 5
""")
csv_query_1_time  = time.time() - start_time

start_time = time.time()
top_hosts_by_bytes = spark.sql("""
SELECT host, SUM(bytes) as byte
FROM web_logs
GROUP BY host
ORDER BY byte DESC
LIMIT 5
""")
csv_query_2_time  = time.time() - start_time


start_time = time.time()
top_urls_by_count = spark.sql("""
SELECT url, COUNT(*) as cnt
FROM web_logs
GROUP BY url
ORDER BY cnt DESC
LIMIT 5
""")
csv_query_3_time  = time.time() - start_time

start_time = time.time()
top_urls_by_bytes = spark.sql("""
SELECT url, SUM(bytes) as byte
FROM web_logs
GROUP BY url
ORDER BY byte DESC
LIMIT 5
""")
csv_query_4_time  = time.time() - start_time
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

[18]:
```
df_parquet.createOrReplaceTempView("web_logs_par")
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

[21]:
```
start_time = time.time()
```

```
df1 = spark.sql("SELECT host, COUNT(url) as cnt FROM web_logs_par GROUP BY host␣
 ↪ORDER BY cnt DESC LIMIT 5")
df1.show()
print("Query 1 Time:", time.time() - start_time)
par_query_1_time  = time.time() - start_time

start_time = time.time()
df2 = spark.sql("SELECT host, SUM(bytes) as byte FROM web_logs_par GROUP BY host␣
 ↪ORDER BY byte DESC LIMIT 5")
df2.show()
print("Query 2 Time:", time.time() - start_time)
par_query_2_time  = time.time() - start_time

start_time = time.time()
df3 = spark.sql("SELECT url, COUNT(*) as cnt FROM web_logs_par GROUP BY url␣
 ↪ORDER BY cnt DESC LIMIT 5")
df3.show()
print("Query 3 Time:", time.time() - start_time)
par_query_3_time  = time.time() - start_time

start_time = time.time()
df4 = spark.sql("SELECT url, SUM(bytes) as byte FROM web_logs_par GROUP BY url␣
 ↪ORDER BY byte DESC LIMIT 5")
df4.show()
print("Query 4 Time:", time.time() - start_time)
par_query_4_time  = time.time() - start_time
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
 ↪layout=Layout(height='25px', width='50%'),...

+-----------------+----+
|             host| cnt|
+-----------------+----+
|     163.206.89.4|7859|
|alyssa.prodigy.com|7099|
|    198.133.29.18|3641|
| bill.ksc.nasa.gov|3122|
|    beta.xerox.com|3064|
+-----------------+----+

Query 1 Time: 6.359786510467529
+--------------------+---------+
|                host|     byte|
+--------------------+---------+
|  alyssa.prodigy.com|177885291|
|        163.206.89.4| 95242337|
|      163.206.137.21| 66541691|
|       198.133.29.18| 61003639|
|cliffy.lfwc.lockh...| 44845440|
+--------------------+---------+

Query 2 Time: 2.162041187286377
+--------------------+-----+
|                 url|  cnt|
+--------------------+-----+
|/images/NASA-logo...|79733|
|/images/KSC-logos...|56696|
|/images/MOSAIC-lo...|49654|
|/images/USA-logos...|49271|
|/images/WORLD-log...|48771|
+--------------------+-----+

Query 3 Time: 1.1771636009216309
+--------------------+----------+
|                 url|      byte|
+--------------------+----------+
|/shuttle/missions...|1058022102|
|/shuttle/missions...| 404609380|
|/shuttle/missions...| 376762426|
|/shuttle/missions...| 374098213|
|/shuttle/missions...| 370688828|
+--------------------+----------+

Query 4 Time: 1.0207579135894775
```

[28]:
```python
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, FloatType

# Initialize Spark session
spark = SparkSession.builder.appName("Query Time Comparison").getOrCreate()

# Data
data = [
    ("Query 1", csv_query_1_time, par_query_1_time),
    ("Query 2", csv_query_2_time, par_query_2_time),
    ("Query 3", csv_query_3_time, par_query_3_time),
    ("Query 4", csv_query_4_time, par_query_4_time)
]

# Define schema
schema = StructType([
    StructField("Query", StringType(), True),
    StructField("CSV Query Time (s)", FloatType(), True),
    StructField("Parquet Query Time (s)", FloatType(), True)
])
```

```
# Create DataFrame
df = spark.createDataFrame(data, schema)

# Display DataFrame
df.show()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:',␣
↪layout=Layout(height='25px', width='50%'),...
```

```
+-------+------------------+----------------------+
|  Query|CSV Query Time (s)|Parquet Query Time (s)|
+-------+------------------+----------------------+
|Query 1|        0.06576061|              6.359855|
|Query 2|        0.02117753|             2.1621094|
|Query 3|        0.02194953|             1.1772327|
|Query 4|       0.017283201|             1.0208356|
+-------+------------------+----------------------+
```

Analysis of Parquet vs CSV Performance in Spark

I finding that querying Parquet files is slower than querying CSV files, it's an unusual situation, as Parquet is generally known for its efficiency in read operations, especially for large datasets. Parquet files are columnar storage formats, which allow for more efficient data compression and encoding schemes. They are optimized for complex queries on large datasets and typically outperform row-based formats like CSV in these scenarios. However, there are several factors that could lead to Parquet files being slower in your case:

1. **Small Dataset Size**: If your dataset is relatively small, the overhead of reading a Parquet file (like file format parsing and schema validation) could be higher than the overhead for a CSV file. CSV files might be faster in cases of small datasets and simple queries.
2. **Columnar Format Overhead**: For certain types of queries, especially those involving a significant portion of the dataset, row-based formats like CSV can sometimes be faster because they don't incur the overhead associated with columnar storage formats like Parquet.
3. **File Size and Partitioning**: If the Parquet file isn't well-partitioned or if it's significantly larger in size compared to the CSV file, it could lead to longer read times.
4. **Execution Plan and Caching**: Spark's execution plan for Parquet files might be less optimized compared to CSV in some scenarios. Additionally, if the CSV data is cached in memory but the Parquet data is not, this could lead to performance differences.
5. **Resource Allocation**: The configuration of your Spark cluster and the resources allocated to your job can significantly impact performance. Make sure your Spark configuration is optimized for the type of workload you're running.
6. **File System and Network Overhead**: The performance can also depend on the underlying file system and network. For instance, reading from HDFS might have different performance characteristics compared to reading from a local file system or a cloud storage service.
7. **Schema Inference**: If you're inferring the schema when reading the CSV files, it might be done once and reused, while reading from Parquet always involves reading the schema.

To investigate further, we can look at the Spark UI to understand the physical and logical plans for your queries. Comparing these plans between the CSV and Parquet queries might provide insights into why there is a performance difference. Additionally, experimenting with different configurations and setups can also help in pinpointing the cause of the slower performance with Parquet files.